

# NGSCB: Threat Models

- **Our Threat Model**
  - No Software-Only Attacks Against RHS
  - No Break-Once/Break-Everywhere (BOBE) attacks
- **No Software-Only Attacks means...**
  - No attacks based on micro-code, macro-code, adapter card scripts, etc.
  - Any attacks launched from the Web or e-mail are "software only"
- **Protection only applies to the release of secrets**

# “Booting” The Nexus

- Nexus is like an OS kernel, so it must boot sometime
- Can boot long after main OS
- Can shut down long before main OS (and restart later)

# NGSCB: Threat Models

- **Our Threat Model**
  - No Software-Only Attacks Against RHS
  - No Break-Once/Break-Everywhere (BOBE) attacks
- **No Software-Only Attacks means...**
  - No attacks based on micro-code, macro-code, adapter card scripts, etc.
  - Any attacks launched from the Web or e-mail are "software only"
- **Protection only applies to the release of secrets**

# HW Keys: Whose are they?

- Answer: The Hardware
  - Used only under explicit user policy.



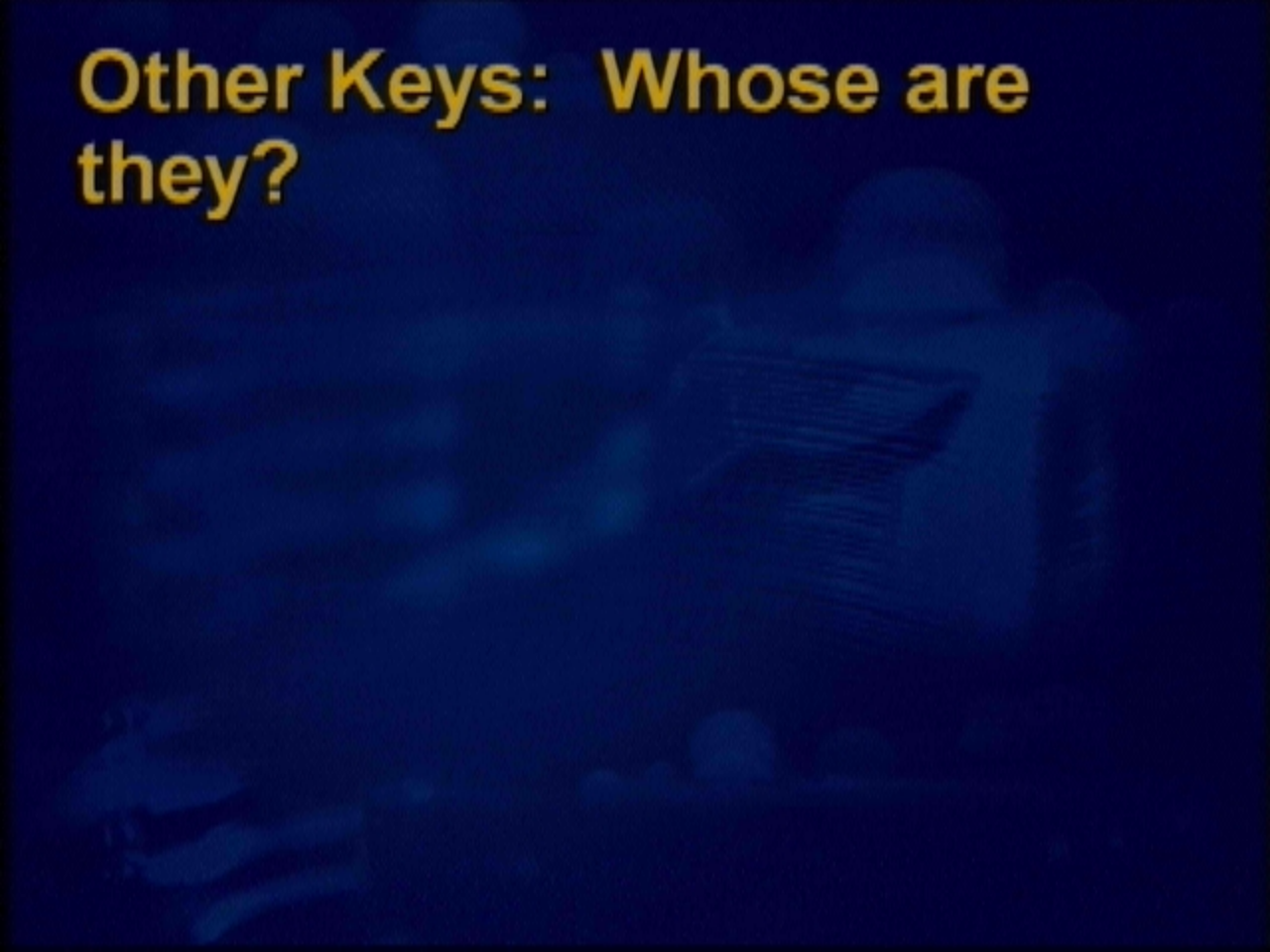
# HW Keys: Whose are they?

- **Answer: The Hardware**
  - Used only under explicit user policy.
- **NGSCB uses two hardware keys directly:**
  - One key is used by Sealed Storage
    - Generated when user "takes ownership"
    - Only available to TPM
    - Randomizing
  - One key is an RSA key used for Attestation
    - Only signs statements like "Nexus with hash x asked me to sign the following statement: y."

# HW Keys: Whose are they?

- **Answer: The Hardware**
  - Used only under explicit user policy.
- **NGSCB uses two hardware keys directly:**
  - One key is used by Sealed Storage
    - Generated when user "takes ownership"
    - Only available to TPM
    - Randomizing
  - One key is an RSA key used for Attestation
    - Only signs statements like "Nexus with hash x asked me to sign the following statement: y."
- **Privacy safeguards built into hardware**
  - Opt-in
  - Disclosure of (public) signing key components is restricted
  - Use of keys in sole control of machine owner

**Other Keys: Whose are they?**



# HW Keys: Whose are they?

- **Answer: The Hardware**
  - Used only under explicit user policy.
- **NGSCB uses two hardware keys directly:**
  - One key is used by Sealed Storage
    - Generated when user "takes ownership"
    - Only available to TPM
    - Randomizing
  - One key is an RSA key used for Attestation
    - Only signs statements like "Nexus with hash x asked me to sign the following statement: y."
- **Privacy safeguards built into hardware**
  - Opt-in
  - Disclosure of (public) signing key components is restricted
  - Use of keys in sole control of machine owner



# Other Keys: Whose are they?

- Answer: Entities authorized by users to access key services
  - User's personal Keys
  - Service provider's Keys
  - Shared Keys

# Machine owner is in complete control

by users to access

5

owner owns nor has access to any

hip is circumscribed and may not even  
entity relying on it.

# Machine owner is in complete control

- Hardware cannot be used without explicit user permission

# Machine owner is in complete control

- Hardware cannot be used without explicit user permission
- No nexus can run without explicit user permission
-



# Machine owner is in complete control

- Hardware cannot be used without explicit user permission
- No nexus can run without explicit user permission
- No NCA can run without explicit user permission

# Machine owner is in complete control

- Hardware cannot be used without explicit user permission
- No nexus can run without explicit user permission
- No NCA can run without explicit user permission
- No NCA can use key services without user permission

# Policies

- Everything that runs today will run on NGSCB systems

# Policies

- Everything that runs today will run on NGSCB systems
- The platform will run any nexus
  - The user will be in charge of what nexuses he chooses to run



# Policies

- Everything that runs today will run on NGSCB systems
- The platform will run any nexus
  - The user will be in charge of what nexuses he chooses to run
- The MS nexus will run any application
  - The user will be in charge of the applications that he chooses to run

# Policies

- Everything that runs today will run on NGSCB systems
- The platform will run any nexus
  - The user will be in charge of what nexuses he chooses to run
- The MS nexus will run any application
  - The user will be in charge of the applications that he chooses to run
- The MS nexus will interoperate with any network service provider

# Misconceptions: NGCSB

- NGSCB will censor or disable content without user permission
  - No policy (except user policy) in NGSCB

# Misconceptions: NGCSB

- NGSCB will censor or disable content without user permission
  - No policy (except user policy) in NGSCB
- NGSCB will lock out vendors
  - No permission (signatures) required to use NGSCB



# Misconceptions: NGCSB

- NGSCB will censor or disable content without user permission
  - No policy (except user policy) in NGSCB
- NGSCB will lock out vendors
  - No permission (signatures) required to use NGSCB
- NGSCB is “super” virus spreader
  - NGSCB applications do not run at elevated privilege

# Misconceptions: NGCSB

- NGSCB will censor or disable content without user permission
  - No policy (except user policy) in NGSCB
- NGSCB will lock out vendors
  - No permission (signatures) required to use NGSCB
- NGSCB is “super” virus spreader
  - NGSCB applications do not run at elevated privilege
- NGSCB NCA is not debuggable
  - Yes it is.

# Misconceptions: TCPA/TCG

- It's the Fritz chip
  - Nope. It's an anti-Fritz chip.

# Misconceptions: TCPA/TCG

- It's the Fritz chip
  - Nope. It's an anti-Fritz chip.
- TCPA/TCG refuses to run unlicensed software
  - Nope. Statement publicly denied by MS, HP and IBM.



# Misconceptions: TCPA/TCG

- It's the Fritz chip
  - Nope. It's an anti-Fritz chip.
- TCPA/TCG refuses to run unlicensed software
  - Nope. Statement publicly denied by MS, HP and IBM.
- Control will be exercised centrally
  - No central authorities required
  - Need for central authorities diminished

# Misconceptions: TCPA/TCG

- It's the Fritz chip
  - Nope. It's an anti-Fritz chip.
- TCPA/TCG refuses to run unlicensed software
  - Nope. Statement publicly denied by MS, HP and IBM.
- Control will be exercised centrally
  - No central authorities required
  - Need for central authorities diminished
- TC will remove effective control of PC from its owner
  - Strengthens owner control

# Nexus: Biquadrants

d-mode" / LHS)

Nexus-Mode (RHS)

- Section 1 of 1

- Process

- M

- UI

User Apps.

Kernel

Main OS

USB

Driver

NexusMgr.sys

HAL

Agent

Agent

Agent

Trusted User  
Engine (TUE)

TSP

TSP

TSP

NCA Runtime Library

Nexus

NAL

Hardware

Secure Input

Secure Video

SSC

CPU

Chipsel

# Nexus: Basic Environment

- Section 1 of Intro to Operating Systems Textbook
  - Process and Thread Loader/Manager
  - Memory Manager
  - I/O Manager
  - Security Reference Monitor
  - Interrupt handling/Hardware abstraction
- But no Section 2
  - No File System
  - No Networking
  - No Kernel Mode/Privileged Device Drivers
  - No Direct X
  - No Scheduling
  - No...
- Kernel mode has no pluggables
  - All of the kernel loaded at boot and in the PCR



# Nexus: Basic Environment

- Section 1 of Intro to Operating Systems Textbook
  - Process and Thread Loader/Manager
  - Memory Manager
  - I/O Manager
  - Security Reference Monitor
  - Interrupt handling/Hardware abstraction
- But no Section 2
  - No File System
  - No Networking
  - No Kernel Mode/Privileged Device Drivers
  - No Direct X
  - No Scheduling
  - No...
- Kernel mode has no pluggables
  - All of the kernel loaded at boot and in the PCR

# Nexus: Basic Environment

- **Virtualization of hardware fundamentals for Agents**
  - Sealed storage, attestation, etc.
- **Minimal Services**
  - Trusted UI Engine
    - XML Based Graphical Services for UI
    - Input Routing/Focus Management
    - Minimum Fonts (inc. Multiple Languages...)
    - Windows Manager
  - IPC
  - TSPs (Trusted Service Provider)
    - Run in User Mode RMS
    - Provide Services
    - Are "Drivers" for Trusted Input/Video

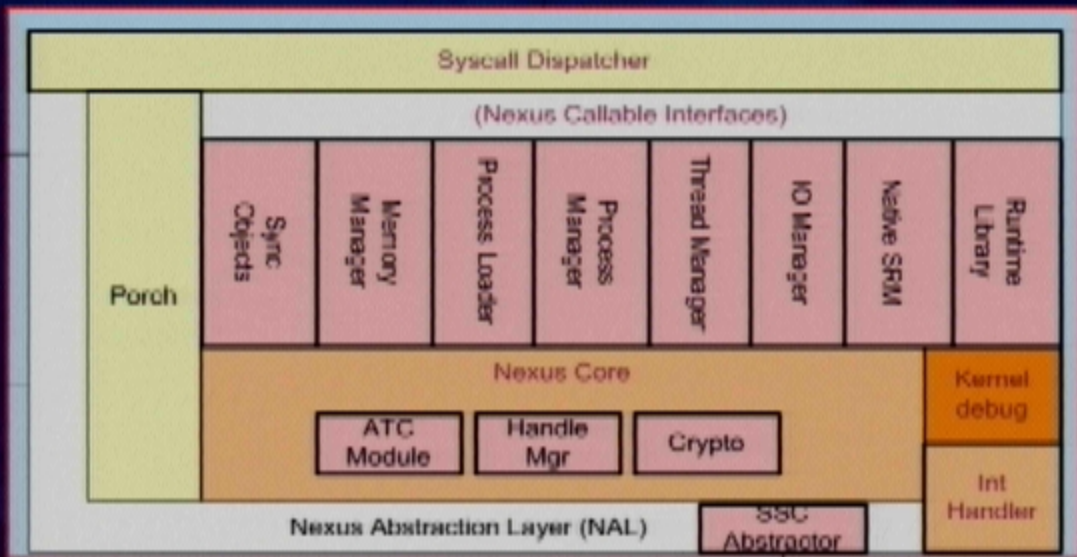
# Close-Up Of Nexus

Nexus.exe

Nx\* Functions

NSS/OS  
Calls

Traps



# Code Identity

- Nexus

- Cryptographic Hash

- Agents

- Manifest (or rather hash of manifest)
  - Debugging Policy
  - Public Key
  - Corresponding Private key authorized to name cryptographic hashes of binaries that identify "this program"
  - Metadata



# Code Identity

- Nexus

- Cryptographic Hash

- Agents

- Manifest (or rather hash of manifest)
  - Debugging Policy
  - Public Key
  - Corresponding Private key authorized to name cryptographic hashes of binaries that identify "this program"
  - Metadata

# User Mode Debugging

- No agents are debuggable without a change in code identity
  - Attestation reflects this change
- Debugging the LHS Shadow Process
  - Uses are different
- Debugging the Agent
  - Context and Read and Write their attestations are different
  - We've redirected RHS
  - Thread control
- Well behaved processes will

# Secret Migration

- Caller gets to specify certain permissions
  - What agents may unseal the set
  - What hardware may unseal the set anywhere
  - What nexus may unseal the set
  - What users may unseal the set (getEnvironment) →
- Agents shouldn't seal
  - They should seal (someSecret) →
    - which seals again
- Backup, restore possible using
  - blob(someSecret),
  - (getEnvironment) → someSecret
  - and cert

# WIIFM: Credential Based Security

- Single simple, flexible, scalable, distributed, credential based security model
  - Programs, use
  - Fine-grained, language
  - General authentication/authorization
- Manageable and
  - Non brittle
  - Administrable
  - Projects Secure
- Framework for policy enforcement
  - Desktop Lockdown
  - Policy assurance (Virus policy, IDS, ...)
- Supports migration of existing Windows security services



# WIIFM: Can Security

- Single simple security

Machine Partitioning  
or “How to keep the TCB  
Small”

# WIIFM: Credential Based Security

- Single simple, flexible, scalable, distributed, credential based security model
  - Programs, users, machines, channels as principals
  - Fine-grained, persistent, declarative claim/assertion/authorization language
  - General authentication and authorization primitives
- Manageable and Flexible
  - Non brittle
  - Administrable
  - Projects Security Perimeter outside Enterprise
- Framework for policy enforcement
  - Desktop Lockdown
  - Policy assurance (Virus policy, IDS, ...)
- Supports migration of existing Windows security services

# WIIFM: Credential Based Security

- Single simple, flexible, scalable, distributed, credential based security model
  - Programs, users, machines, channels as principals
  - Fine-grained, persistent, declarative claim/assertion/authorization language
  - General authentication and authorization primitives
- Manageable and Flexible
  - Non brittle
  - Administrable
  - Projects Security Perimeter outside Enterprise
- Framework for policy enforcement
  - Desktop Lockdown
  - Policy assurance (Virus policy, IDS, ...)
- Supports migration of existing Windows security services

# Overview

- Requirements
- System Architecture



# **Machine Partitioning or “How to keep the TCB Small”**

# Overview

- Requirements
- System Architecture

# Mass Market Requirements

- **Open architecture**
  - Decentralized production of HW peripherals
  - Any SW runs
- **Legacy**
  - No fundamental changes to PC platform hardware
  - No fundamental changes to OS software (Windows)
  - Most peripheral hardware must work without changes
- **Performance**
  - Security features should not deteriorate performance significantly (at most a few percent).
- **Cost**
  - No significant increase in the cost of PC platform hardware

# Security Requirements

- Programs must be able to receive, process, output data without leaking or subversion
  - Confidentiality
  - Integrity
- Not necessarily availability
- Need high assurance for these properties
  - Should not depend on BIOS, option ROMS, device drivers, large operating systems



# System Architecture



# Existing Approaches

- **Traditional high assurance systems**
  - Often Orange Book as the main guideline
  - Examples
    - SE/VMS, SCOMP, KSOS, KVM/370
  - Not designed for today's mass market requirements
- **Add security features to a mass market OS**
  - Example: SE/Linux
  - What about assurance, given
    - The large size of the OS
    - The need to optimize for performance and features
    - The diversity and complexity of device drivers
      - 1.5 million lines of code per video driver
      - Several manufacturers; many models; frequent updates

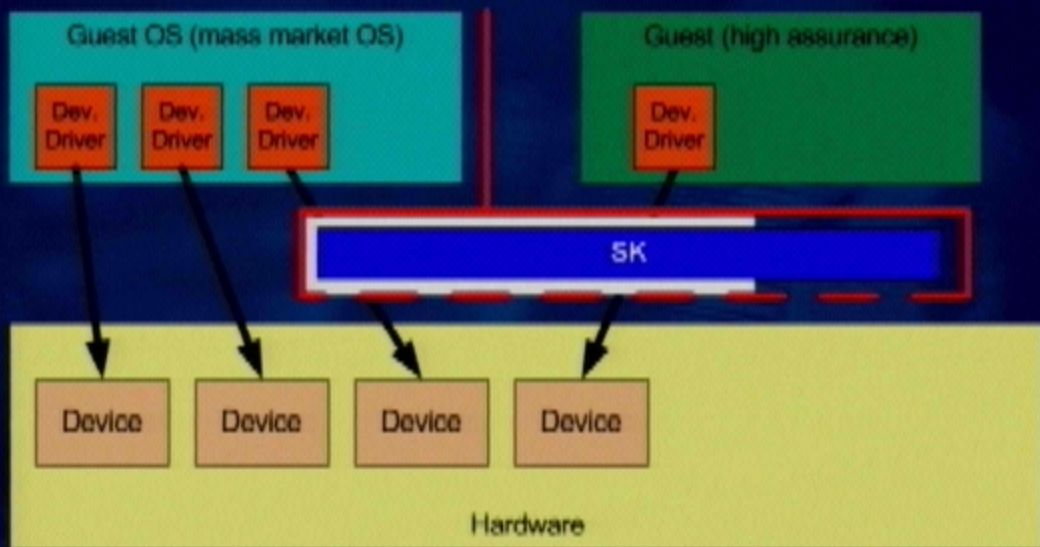
# Dual System Approach

- **Combine a mass market system and a high assurance system in one computer**
  - Examples: Perseus, Secure Coprocessors
- **Software implementations**
  - Base layer hosts and isolates mass market OS and high assurance OS.
    - Base layer could be a VMM, an exokernel, a microkernel (Perseus)
- **Requirements for the base layer**
  - Must isolate hosted OSs (each OS can have exclusive access to some resources (pages of physical memory, keyboard))
  - High assurance
- **Many systems put all DMA device drivers into the base layer.**
  - Low assurance

# NGSCB Overview

- Dual System
- Software isolation layer (security kernel – SK)
- SK hosts and isolates guest OSs
- SK assigns devices (including DMA) to guest OSs
  - SK does not manage DMA devices
  - SK does not contain DMA device drivers
  - Reminiscent of Exokernels – but existing Exokernels manage DMA devices.
- SK exposes VMM-style machine interface for one guest OS (mass market OS – Windows).
  - Reminiscent of V=R mode on IBM mainframes.





# The Rest of this Section

- How the SK manages DMA devices
- How the SK implements isolation
- Prototype implementation

# DMA: Device Problem

- We allow guest OSs to program DMA devices directly.
- But: Any guest OS in control of a DMA device
  - Can program the DMA controller
  - Obtain unrestricted access to all of memory by means of DMA.
- Need a new HW mechanism to restrain DMA devices.
- SCOMP followed a similar approach.

# DMA: Hardware Access Control

- For physical memory and memory mapped devices
- SK can specify for each page if DMA devices are allowed to access it.
  - 1 bit per page
- Enforced by the memory controller
- Limited expressive power
  - Trade off between complexity and functionality



# DMA: Management (in theory)

- The SK assigns devices to guest OSs for their exclusive use.
  - Memory mapped devices: Allow the guest OS to map the device into its address space.
  - Exokernel-style “secure binding”
- Interrupts
  - SK receives interrupts.
  - SK dispatches device interrupts to the guest OS that owns the device.
  - Guest OS handles interrupt.
- In practice (at least initially)
  - SK may not be able to control interrupt handling.
  - Static allocation of most devices to Windows.

# Isolation

- **Requirements**

- Enable guest OSs to have exclusive access to resources (memory, devices)
- Expose native hardware interface to at least one guest OS
- Related to VMM requirements

- **Why not just build a VMM?**

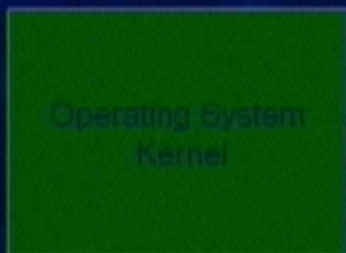
- Different treatment of devices (we export; VMMs virtualize)
- Existing VMMs work great for most tasks. But
  - Some things do not work (device access).
  - Some things are slow.
- x86 processors are not virtualizable. Building a VMM for the x86 is tricky.

- **We introduce a new hardware assist: Ring -1.**

Ring 3



Ring 0



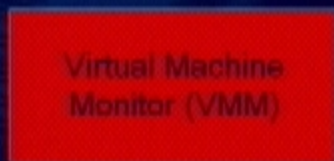
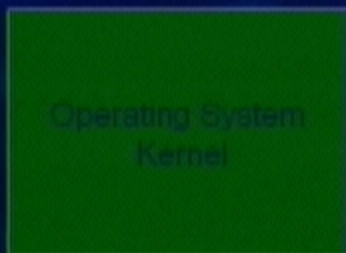
Native operation

Traditional VMM

Ring 3



Ring 0



Native operation

Traditional VMM



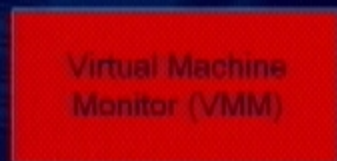
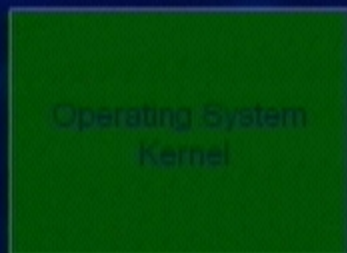
Ring 3



Operating  
System  
Kernel

Operating  
System  
Kernel

Ring 0



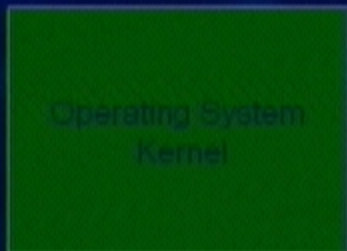
Native operation

Traditional VMM

Ring 3



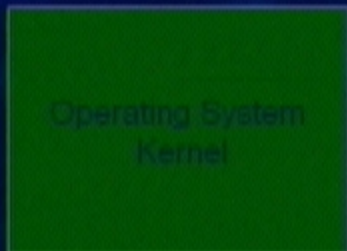
Ring 0



Ring 3



Ring 0

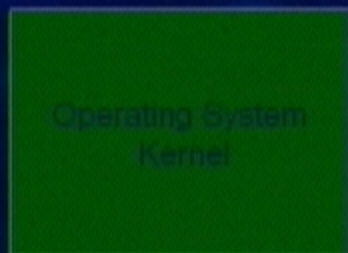


Ring -1

Ring 3



Ring 0

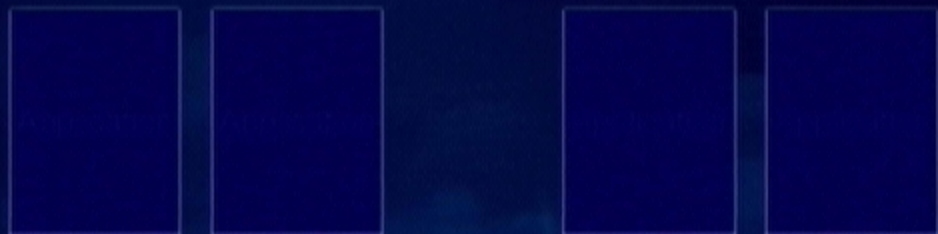


Ring -1





Ring 3



Ring 0



Ring -1



# Memory Isolation

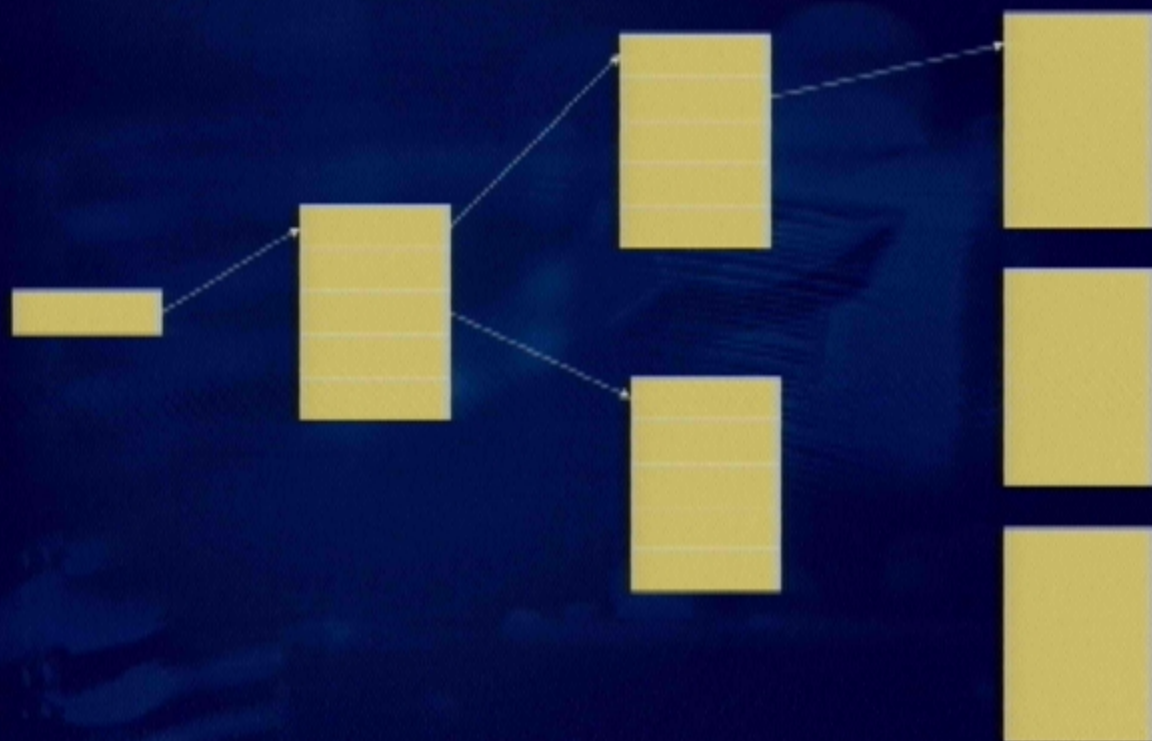
- Every VMM has to do this in some form.
  - Classical algorithm: shadow page tables
  - V=R optimization for Windows “guest”
  - Further optimizations based on specific behavior of the Windows memory manager
- The SK grants a guest OS access to a resource by allowing it to map it.
- Every guest OS instruction that could change the virtual-to-physical page map traps into the SK
  - Write to CR3
  - Write using read-only mapping
    - The SK enforces that all mappings to page tables and page directories are read-only

PDBR /  
CR3

Page  
Directory

Page  
Table

Other  
Pages

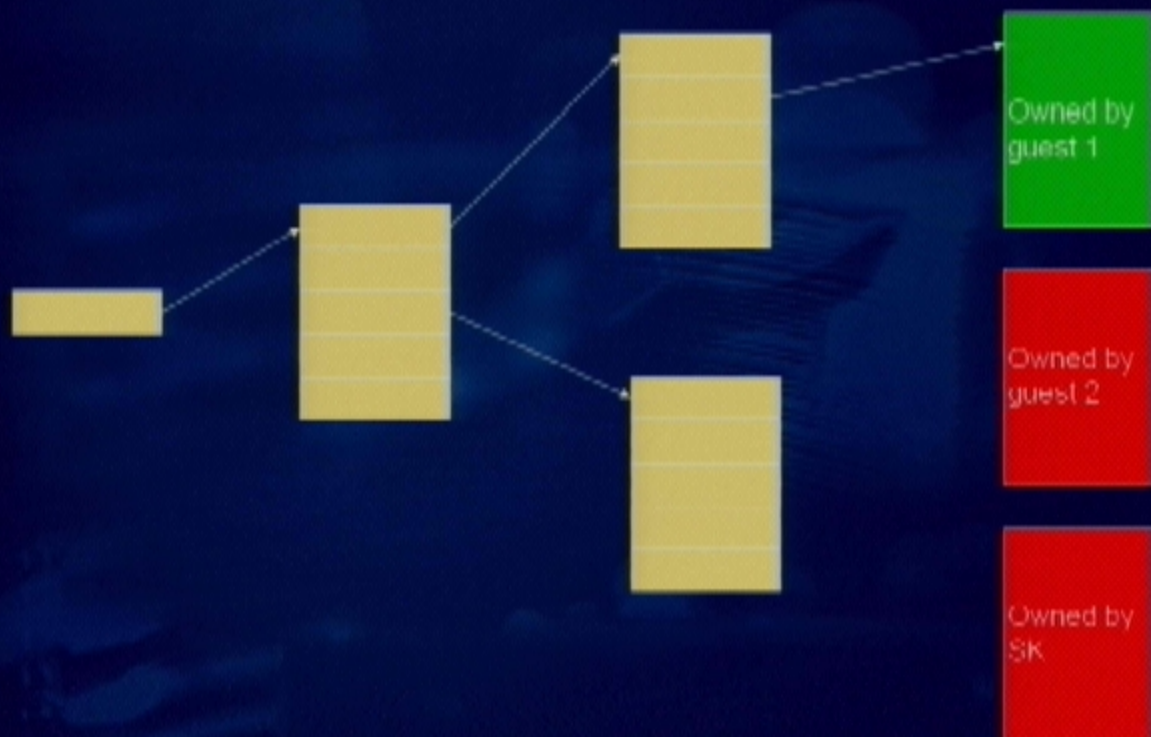


PDBR /  
CR3

Page  
Directory

Page  
Table

Other  
Pages



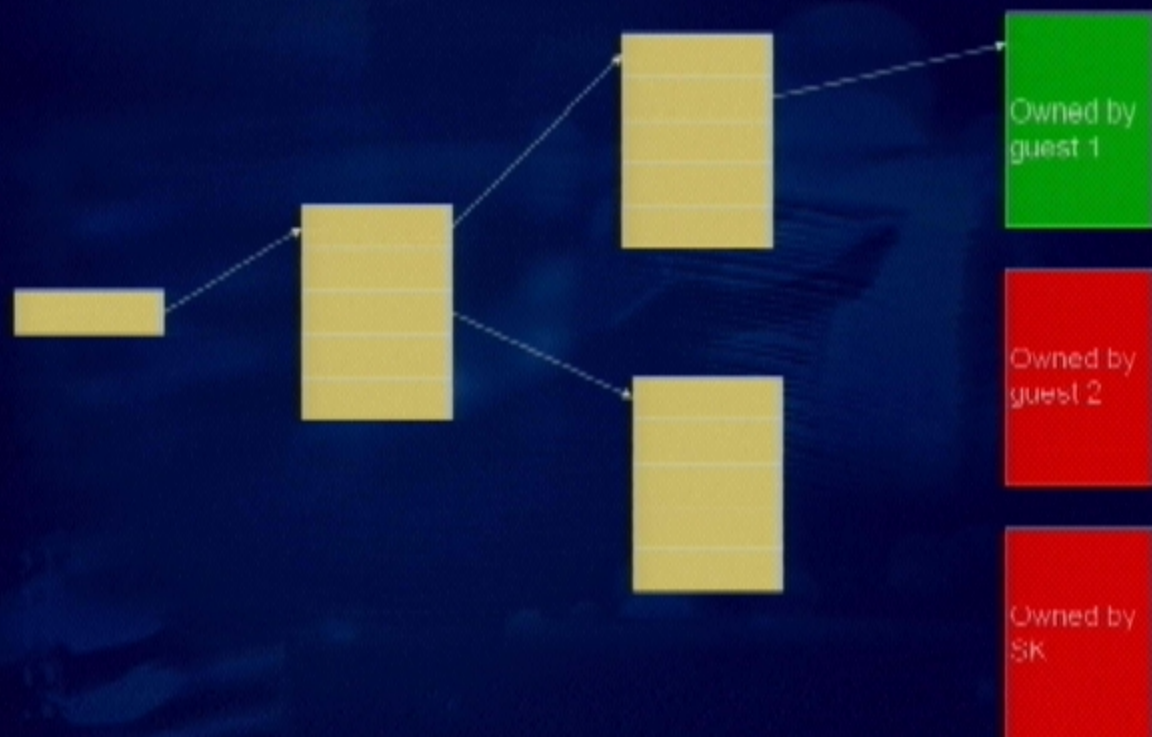


PDBR /  
CR3

Page  
Directory

Page  
Table

Other  
Pages

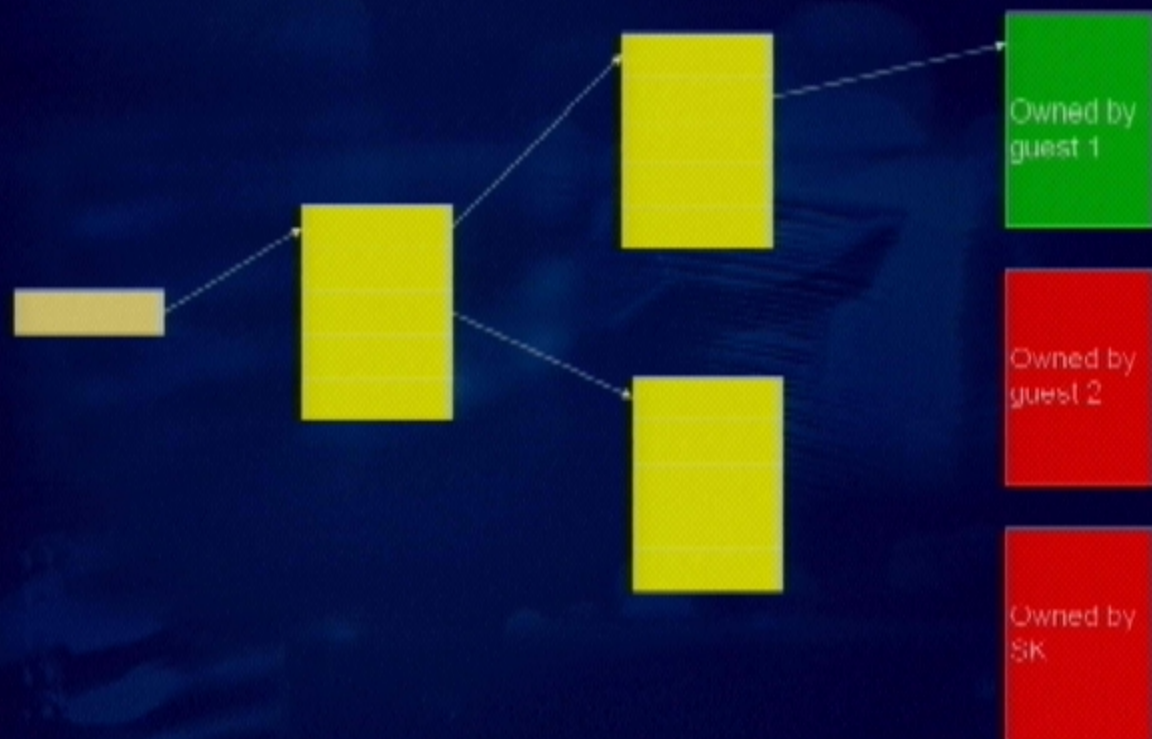


PDBR /  
CR3

Page  
Directory

Page  
Table

Other  
Pages

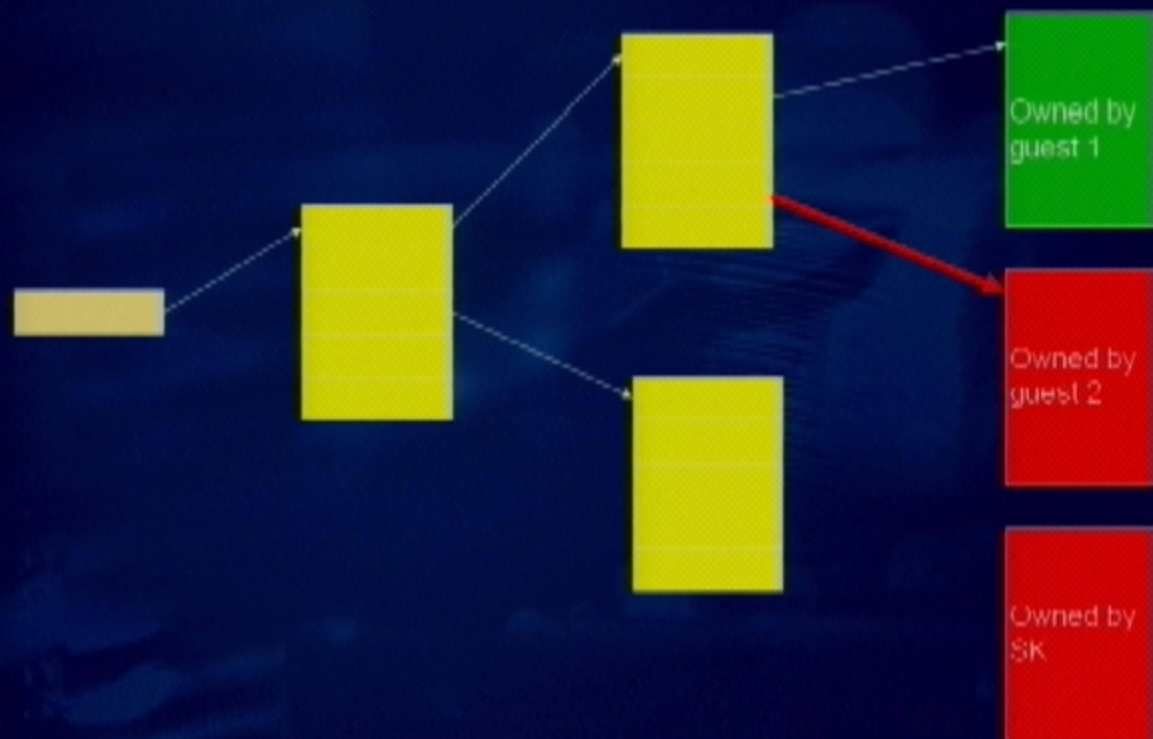


PDBR /  
CR3

Page  
Directory

Page  
Table

Other  
Pages

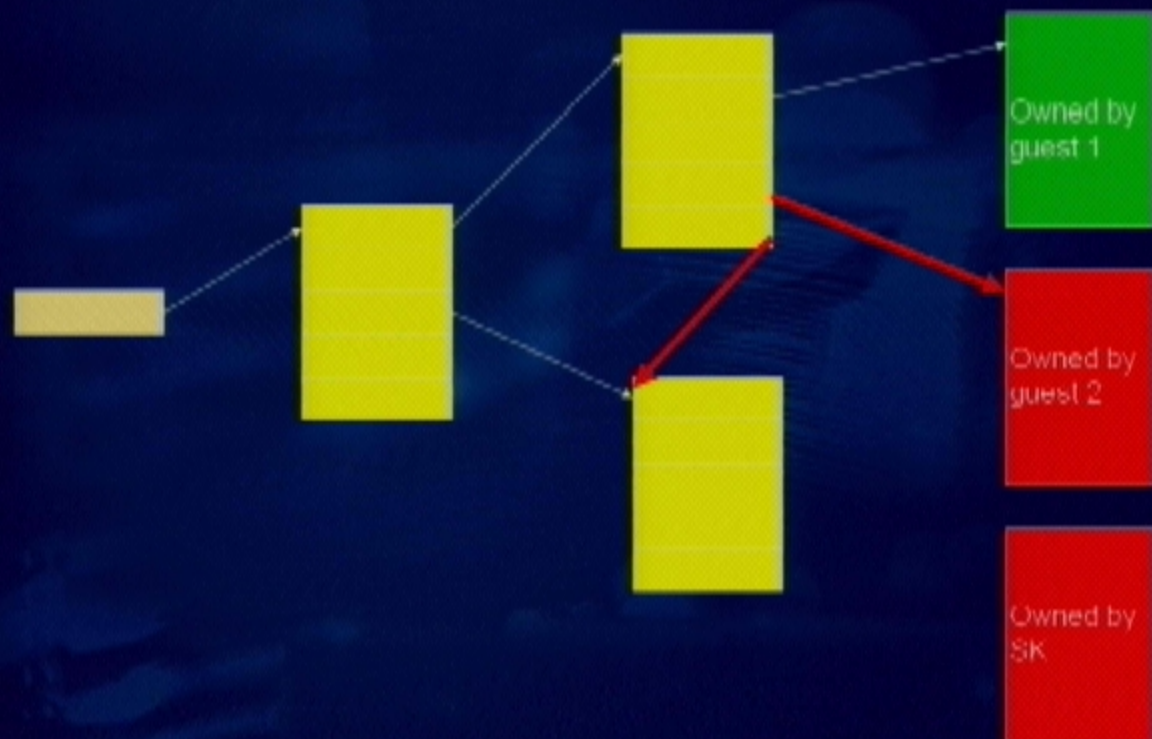


PDBR /  
CR3

Page  
Directory

Page  
Table

Other  
Pages



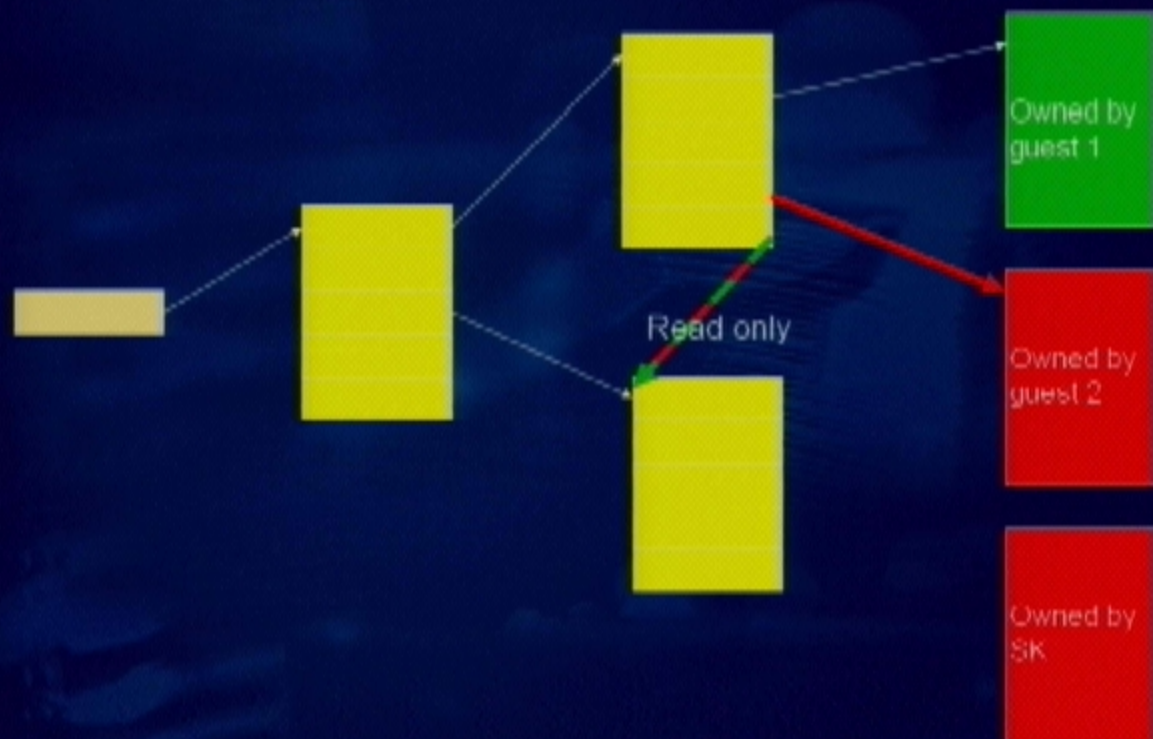


PDBR /  
CR3

Page  
Directory

Page  
Table

Other  
Pages



# Isolation: Optimization

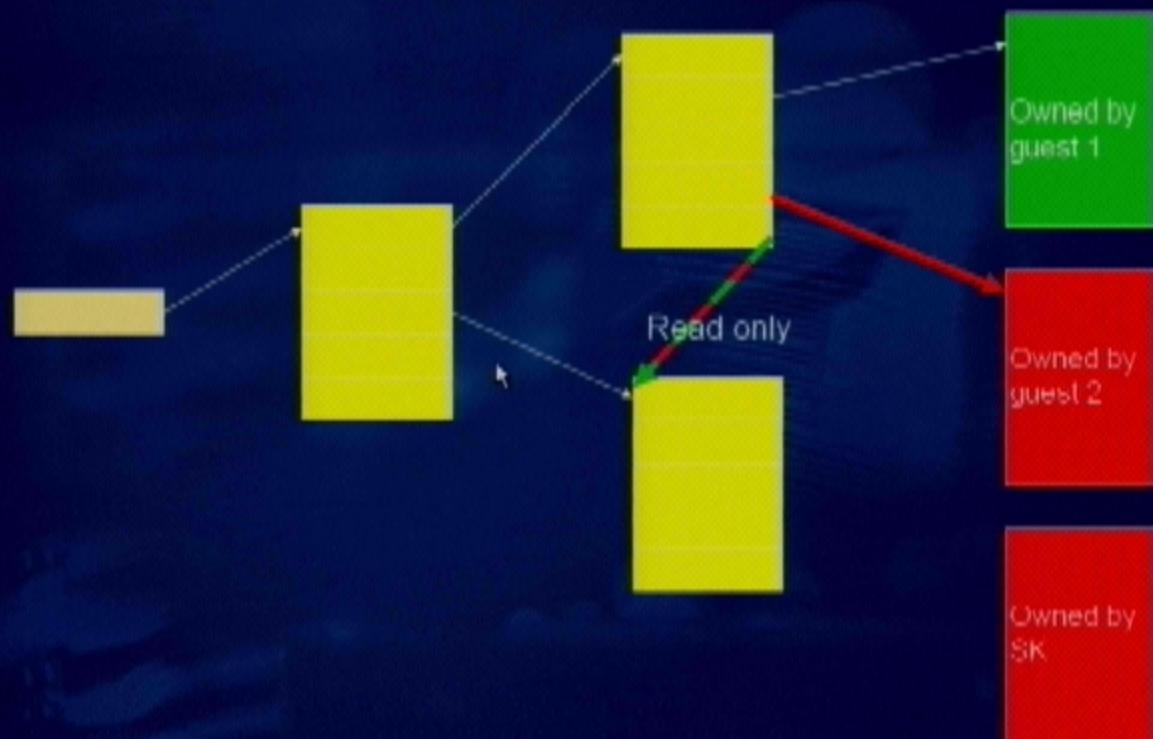
- Use caching to reduce the average cost of trap handling:
  - D1: pages that are safe to be used as page directories
    - On Cr3 load: If the new page directory is in D1, no further checking is necessary
  - D2: pages that are safe to be used as page tables
    - Speeds up certain PD edits
  - Several reference counters

PDBR /  
CR3

Page  
Directory

Page  
Table

Other  
Pages



# Isolation: Optimization

- Use caching to reduce the average cost of trap handling:
  - D1: pages that are safe to be used as page directories
    - On Cr3 load: If the new page directory is in D1, no further checking is necessary
  - D2: pages that are safe to be used as page tables
    - Speeds up certain PD edits
  - Several reference counters



# SK Prototype

- Implemented isolation and hosting components of the SK
- Test bed:
  - Unmodified PC hardware
  - Instrumented version of Windows XP
    - Write to CR3 → trap to SK
    - Write fault → trap to SK
    - Process destruction → notify SK
    - Delay loops and artificial TLB flushes to account for HW differences
- Goals
  - Check for surprises in the behavior of Windows.
  - Check that our validation methodology is feasible.
  - Test performance; gather data for optimizations.

# Prototype Implementation

- About 2000 lines of C code
- Core PTEC algorithm: about 1000 lines
  - Formally specified in first order logic
  - Verified using a theorem prover
  - C code derived from formal specification
- Prototype is not optimized!
- Test Bed for experiments
  - 2.5 GHz Pentium 4
  - 1 GByte DDR RAM
  - nVidia GeForce2 graphics card; WD400BB hard disk

# Event counts

Event	Count	Avg. time ( $\mu$ s)
CR3 load (new)	63	2006.3
CR3 load (from D1 cache)	971,391	0.234
PD edit	8,860	19.726
PT edit	9,082,535	1.58

Events counted over one run of the Content Creation  
Winstone 2002 benchmark program.

# Micro Benchmarks

Benchmark	WinXP native	SK no delay	SK 1000 delay cycles TLB Flush	VMware
Disk I/O (MB/s)	40.87	40.62	40.67	25.46
DirectDraw (frames/s)	18.64	18.66	18.64	Failed
GDI line draw (ms)	1.65	2.40	2.95	14.15
Context Switch (ms)	4.28	5.51	6.86	28.34
Memory Allocation (ms)	1.27	11.65	12.97	10.80
Process Creation (ms)	2.70	7.00	7.76	7.61



# Synthetic Benchmarks

Benchmark	WinXP native	SK no delay	SK 1000 delay cycles	VMWare
Business Winstone 2002	25.7	24.5	24.2	22.5
Content Creation Winstone 2002	35.0	33.6	33.4	27.9

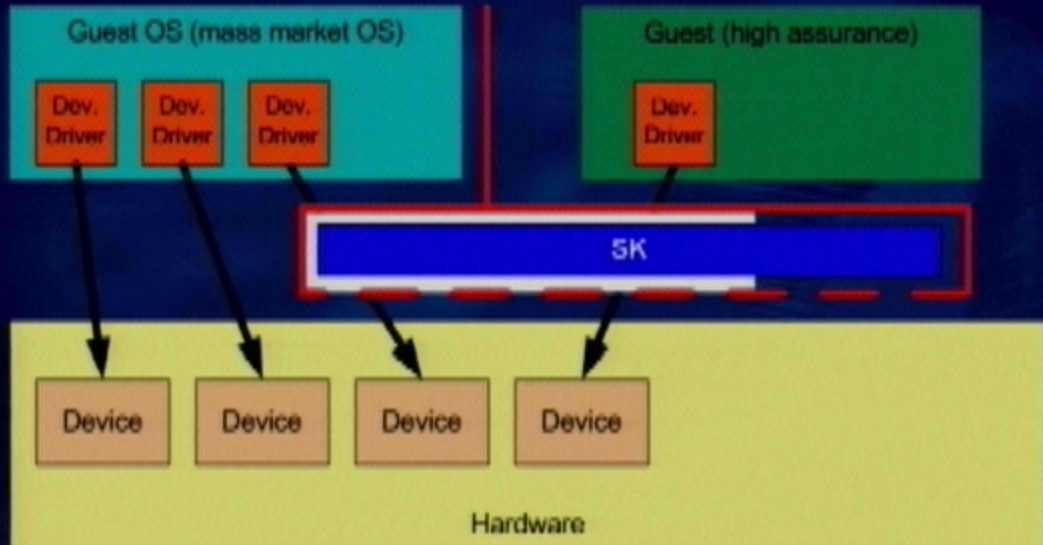
**Business Winstone:** Word, Excell etc.

**Content Creation Winstone:** Adobe Photoshop, Adobe Premiere,  
Windows Media Encoder, Netscape Webbrowser

# Position Check

- **Dual System**
  - One mass market guest
  - One high assurance guest
- **Very small isolation layer (SK)**
  - Fast, verifiable, compatible with legacy
  - Device management left to guests
  - No complicated virtualization schemes
- **Require changes to chipset and CPU**
  - HW enforced access restrictions for DMA devices
  - Ring -1 and privileged instructions

# The High Assurance Guest



# Micro Benchmarks

Benchmark	WinXP native	SK no delay	SK 1000 delay cycles TLB Flush	VMware
Disk I/O (MB/s)	40.87	40.62	40.67	25.46
DirectDraw (frames/s)	18.64	18.66	18.64	Failed
GDI line draw (ms)	1.65	2.40	2.95	14.15
Context Switch (ms)	4.28	5.51	6.86	28.34
Memory Allocation (ms)	1.27	11.65	12.97	10.80
Process Creation (ms)	2.70	7.00	7.76	7.61